LINKÖPING UNIVERSITY
Institutionen för datavetenskap
Patrick Lambrix / Olaf Hartig / Valentina Ivanova / Zlatan Dragisic

Tentamen

DF22300 Advanced Data Models and Databases 2016

Grades:

For a pass grade all questions need to be answered and almost all correctly answered.

Instructions:

- This is an *individual* exam. If you have questions, ask the course leader.
- Send your answers to: Patrick Lambrix, Institutionen för datavetenskap, Linköpings universitet, 581 83 Linköping.
- Deadline: February 1, 2017.
- Write a contact e-mail address on your hand-in.
- If there are references to papers in the questions, then links to the papers will be on the course home page.

Question 1: Information Retrieval

Assume that we have two documents in our document base. Document 1 contains 'enzyme' 5 times, 'gene' 10 times, 'protein' 0 times and 'signal' 8 times. Document 2 contains 'enzyme' 0 times, 'gene' 0 times, 'protein' 7 times and 'signal' 1 time.

(i) Assume that we use the vector model for information retrieval. Assume that we are only interested in the words gene, enzyme, protein and signal.
1. Explain tf and idf in the vector model.
2. In which cases is a weight w_{ij} in a document vector equal to 0?
3. Give the document representations for Document 1 and Document 2 according to the tf-idf model.

(ii) Assume that we use the boolean model for information retrieval. Assume that we are only interested in the words gene, enzyme, protein and signal.
1. Give the document representations for Document 1 and Document 2 according to the Boolean model.
2. Represent the query for all documents containing gene or protein, but not signal. Compute then the completed DNF (disjunctive normal form) of the query.

Question 2. Semi-structured data

(i) Below, a (simplified) description of the biological data source PIR-PSD, is given. This data source contains information about protein sequences. Each entry in the data source has information about the entry (identification numbers, dates of creation and update), names of the protein sequence and the parts that it contains, the organism from which the sequence was taken, references to the literature describing the sequence, references to other related databases and the protein sequence. The simplified PIR-PSD schema is given as a DTD. Draw a strong data guide for PIR-PSD. Use the OEM model.

```
<!-- ********************************************************************
PIR-International Protein Sequence Database (PSD)
******************************************************************** -->

<!-- ProteinEntry: the root element. -->
<!ELEMENT ProteinEntry (header,protein,organism,reference*,sequence)>

<!-- header: database information. -->
<!ELEMENT header (uid,accession*,created_date,seq-update_date)>

<!ELEMENT accession (#PCDATA)> <!-- accession number -->
<!ELEMENT created_date (#PCDATA)> <!-- date (DD-MMM-YYYY) -->
<!ELEMENT seq-update_date (#PCDATA)> <!-- date (DD-MMM-YYYY) -->

<!-- protein: the protein-names. -->
<!ELEMENT protein(name,alt-name*,contains*)>

<!ELEMENT name (#PCDATA)> <!-- protein name -->
<!ELEMENT alt-name (#PCDATA)> <!-- alternate protein name -->
<!ELEMENT contains (#PCDATA)> <!-- activity name -->

<!-- organism: identification of the biological source. -->
<!ELEMENT organism (source,common-name?,note*)>

<!ELEMENT source (#PCDATA)> <!-- source name -->
<!ELEMENT common-name(#PCDATA)> <!-- common name -->
```

```
<!-- reference -->
<!ELEMENT reference (refinfo,note*)>

<!-- refinfo: identification of the literature source. -->
<!ELEMENT refinfo (authors,citation,title?,xrefs?)>

<!ELEMENT authors (author+)> <!-- list of authors -->
<!ELEMENT author (#PCDATA)> <!-- author name -->
<!ELEMENT citation (#PCDATA)> <!-- citation name -->
<!ELEMENT title (#PCDATA)> <!-- title text -->

<!ELEMENT xrefs (xref+)> <!-- cross-references -->
<!ELEMENT xref (db,uid)> <!-- a cross-reference -->
<!ELEMENT db (#PCDATA)> <!-- database tag -->

<!-- sequence: the amino acid sequence. -->
<!ELEMENT sequence (#PCDATA)> <!-- amino acid symbols and
 punctuation -->

<!-- General elements. Elements that can be contained in several
 other elements. -->
<!ELEMENT note (#PCDATA)> <!-- note text -->
<!ELEMENT uid (#PCDATA)> <!-- entry identifier -->
```

(ii) In figures 1 and 2 (simplified and slightly modified) data from a SwissProt data source is given. (The dashed nodes in the figures refer to nodes in the other figure. Both figures together make up the data source.) Draw a strong data guide for this data. Use the OEM model.

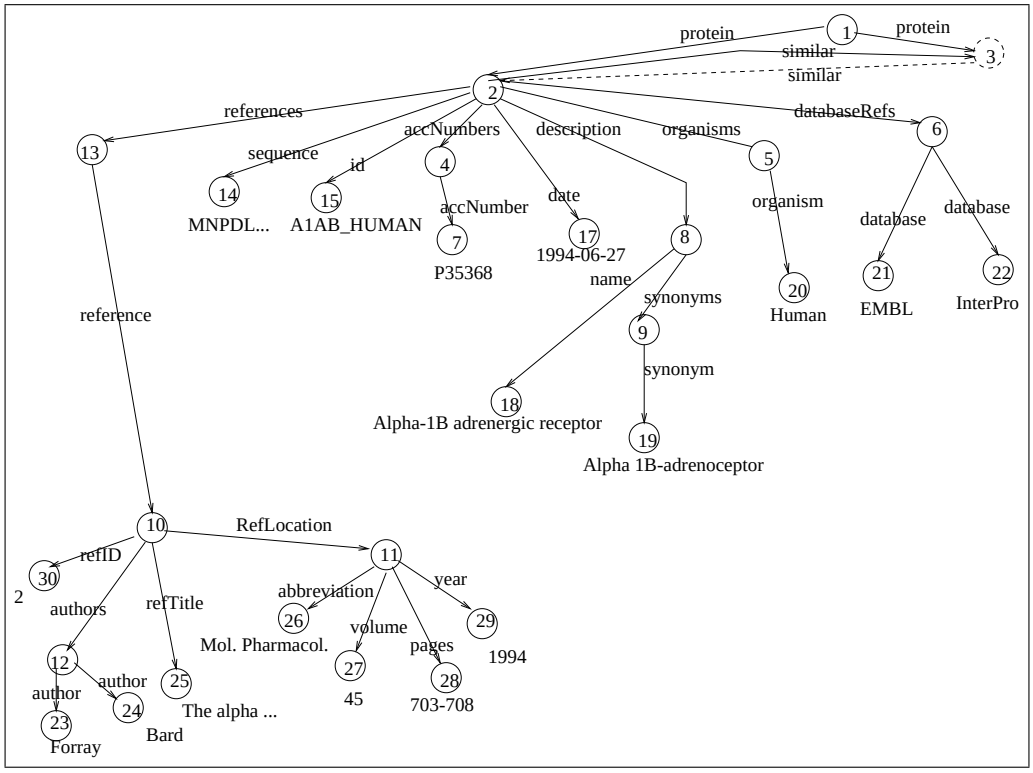

Figure 1: SwissProt db -1

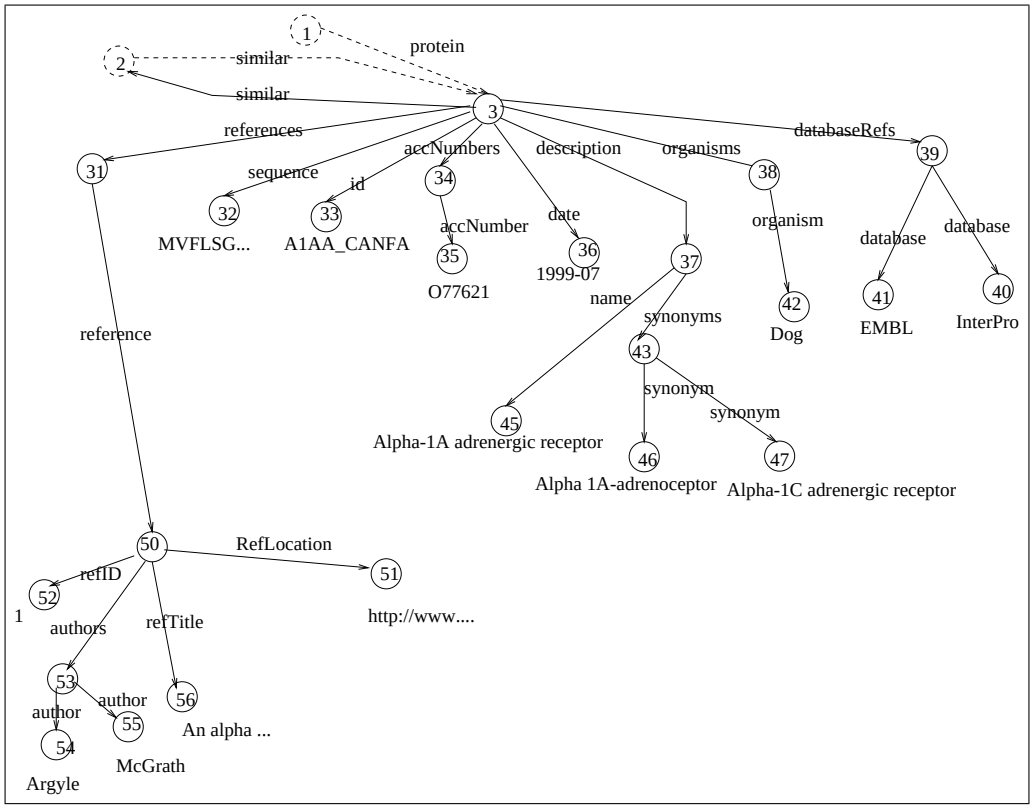

Figure 2: SwissProt db -2

Question 3: Description Logics

Define the following concepts using description logics:

C1: animals that have at least 4 legs and all foods they eat are wheat plants
C2: animals that have at least 2 legs and all foods they eat are plants

Does C2 subsume C1, i.e. C1 isa C2? Prove your answer using a tableau algorithm. (Use the Baader, Nutt chapter in the reading list.)

Question 4: DB vs KB

Describe the difference between open-world assumption and closed-world assumption. Give examples.

Question 5: Ontology Engineering

(i) Give 4 different kinds of matchers for ontology alignment. For each kind of matcher give an example and explain briefly what it does.

(ii) Given the following axioms in an ontology: $A \sqsubseteq B$; $A \sqsubseteq E$; $B \sqsubseteq C$; $B \sqsubseteq D$; $C \sqsubseteq F$; $D \sqsubseteq F$; $F \sqsubseteq G$; $E \sqsubseteq G$; $A \sqsubseteq \neg G$.

Debug the ontology. Compute MIPS and MUPS.

Question 6. XML

Consider an XML document 'data.xml' that is valid for the DTD in Question 2(i).

(i) Write an XPath expression that selects the citation names of all literature sources that have a cross-reference to a database called DBLP.

(ii) Write an XPath expression that selects every reference of which the last author is Adam Smith.

(iii) Write an XQuery expression that lists the accession numbers of all protein entries that have the same creation date as some entry for the protein with accession number NM_000784. Accession number NM_000784 must not be in the list.

Question 7. RDF and SPARQL

Consider the following set of RDF triples.

```
ex:cid651  rdf:type  ex:CourseOccasion .
ex:cid651  ex:code   "TDDD12" .
ex:cid651  ex:year   2014 .
ex:cid651  ex:instructor   ex:AdamSmith .

ex:cid337  rdf:type  ex:CourseOccasion .
ex:cid337  ex:code   "TDDD12" .
ex:cid337  ex:year   2017 .

ex:cid810  rdf:type  ex:CourseOccasion .
ex:cid810  ex:code   "TDDD37" .
ex:cid810  ex:year   2014 .
ex:cid810  ex:instructor   ex:EvaHegen .

ex:cid810  rdf:type  ex:CourseOccasion .
ex:cid810  ex:code   "TDDD37" .
ex:cid810  ex:year   2013 .
ex:cid810  ex:instructor   ex:AdamSmith .

ex:AdamSmith  ex:name  "Adam Smith" .
ex:EvaHegen   ex:name  "Eva Hegen" .
```

(i) What is the result of evaluating the following SPARQL query (prefix declarations omitted) over the given set of RDF triples? Represent the result in a tabular form.

```
SELECT ?y ?i WHERE {
  ?c  ex:code  "TDDD37" .
  ?c  ex:year  ?y .
  ?c  ex:instructor  ?i .
}
```

(ii) What is the result of evaluating the following SPARQL query (prefix declarations omitted) over the given set of RDF triples? Represent the result in a tabular form.

```
SELECT ?y ?n WHERE {
  ?x  ex:code  "TDDD12" .
  ?x  ex:year  ?y .
  OPTIONAL {
    ?x  ex:instructor  ?i .
    ?i  ex:name  ?n .
  }
}
```

(iii) Write a SPARQL query for RDF data such as the triples given above to retrieve the URI of every course occasion before 2015.

(iv) Write a SPARQL query for RDF data such as the triples given above to retrieve the name of every person who was instructor for multiple occasions of the same course (i.e., different course occasions with the same course code).

Question 8. NoSQL databases

Explain how the MapReduce programming model works. Show how it could be applied to the problem of indexing documents (write pseudo code, inputs and outputs for the map and reduce steps).

Question 9. Graph databases

Consider the following vertex compute function:

```
void compute(Vertex v) {
  if ( isFirstSuperStep() ) {
    v.setValue( v.getVertexID() );
    v.sendMessageToAllNeighbors( v.getVertexValue() );
  }
  else {
    minValue = v.getVertexValue();
    for ( msg <= v.getIncomingMessages() ) {
      if ( msg < minValue )
        minValue = msg;
    }

    if ( minValue < v.getVertexValue() ) {
      v.setValue( minValue );
      v.sendMessageToAllNeighbors( v.getVertexValue() );
    }
  }

  v.voteToHalt();
}
```

This vertex compute function presents a vertex-centric approach to compute weakly connected components (WCC) of a directed graph. Note that the method 'sendMessageToAllNeighbors' sends the given message to both all out-neighbors and all in-neighbors of the vertex.

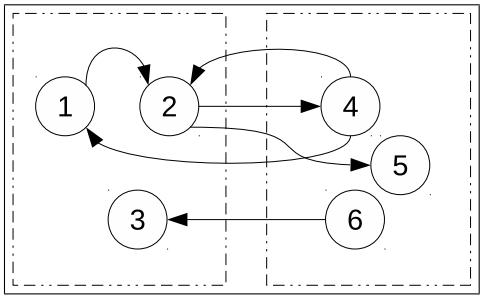

Figure 3: Graph

(i) Consider the graph in Figure 3 (ignore the dashed rectangles for the moment). The numbers in the vertexes denote the vertex identifiers. A bulk synchronous parallel (BSP) execution of the aforementioned vertex compute function over the given graph will terminate after 3 supersteps. Fill the following table with the vertex values that each vertex of the graph has after each superstep.

```
         | step 1 | step 2 | step 3 |
-------------------------------------
vertex 1 |
vertex 2 |
vertex 3 |
vertex 4 |
vertex 5 |
vertex 6 |
```

(ii) Recall that combiners are associative, commutative functions that combine two messages into one. Fill in pseudo-code into the following function such that this function can be used as a combiner for the vertex-centric WCC algorithm given above.

```
Integer combine( Integer msg1, Integer msg2 ) {
  // your code goes here:
}
```

(iii) Consider a partitioning P= P1,P2 of the example graph into two disjoint, equally sized vertex sets: P1 = 1,2,3 and P2 = 4,5,6 . The dashed rectangles in the figure illustrate this partitioning. Assume the two partitions are assigned to two different worker nodes in a compute cluster. Specify another partitioning of the example graph into two disjoint, equally sized vertex sets such that, when your two partitions are assigned to two different worker nodes in a compute cluster, an execution of a vertex-centric algorithm (such as the WCC algorithm given before) will cause less network traffic than the aforementioned partitioning P.

Question 10. Integration of data sources

(i) *Global model*

Draw a global domain model based on your data guides from question 1. Use the information in figure 4 about which concepts in SwissProt match to concepts in PIR-PSD. Assume that concepts with the same name in the two data sources match. When labeling, give priority to SwissProt terminology.

(ii) *Global as view and local as view*

Discuss the global as view approach versus the local as view approach regarding the mappings between the global model and the content of the local sources as well as regarding query processing. *Exemplify* using examples based on the scenario in question 1 and question 3(a). (If needed, you may extend the scenario or the data source descriptions.)

SwissProt concept	PIR-PSD concept
id	uid
accNumbers	accession
date	created_date
description	protein
synonym	alt_name
refTitle	title
refLocation	citation
databaseRefs	xrefs
database	db

Figure 4: Matching concepts.